All topics from class and homework may be covered, including those not listed here. This handout just gives a list of topics covered since the first midterm, on which the midterm will concentrated, but any topic from the course is possible for the test.

- Binary Search and Variants
- Graphs representations
- BFS and DFS (including edge type labels and discovery/finish times)
- Topological sort by DFS and without DFS
- Strongly connected components
- Prim’s and Kruskal’s minimum spanning tree programs
- Three single-source shortest path algorithms
- Disjoint Set (or Union-Find) data structure with union-by-size and path compression
- Floyd-Warshall all-pairs shortest-paths
- Material from the first midterm is also fair game, but this test will definitely concentrate on material since the first midterm.

1. Consider the following graph $G$ given in ingraph notation:

   $G = (V, E)$
   $V = \{1, 2, 3, 4, 5, 6\}$
   $E = \{1 : 3, 4, 5; 2 : 1, 5; 3 :, 4 :, 5 :, 6 : 1, 3, 5; \}$

   (a) Give the outgraph notation for $G$.
   (b) Give the indegrees and outdegrees for the vertices of $G$.
   (c) Give a topological ordering of the vertices of $G$.

2. Consider the following graph, given in outgraph notation:

   $G = (V, E)$
   $V = \{1, 2, 3, 4, 5, 6\}$
   $E = \{1 : 3, 4, 5; 2 : 1, 5; 3 :, 4 :, 5 :, 6 : 1, 3, 5; \}$

   Run DFS on $G$. Assume that the main loop of DFS goes through vertices in the given order, and that within each node, neighbors are visited in the order given for each node.

   (a) Give start and finish times for each node.
   (b) Label each edge as a tree-edge, back-edge, forward-edge, or cross-edge.
   (c) Give the parenthesis labeling for this DFS.
   (d) Label strongly connected components in this graph.

3. Consider the following graph (Some graph would go here. If you like, consider the undirected version of the graph from the previous problem, where the weight of edge $(u, v) = u + v$):

   (a) Run Kruskal’s algorithm on this graph, and give the order in which edges are added to the MST.
(b) Run Prim’s algorithm on this graph, and give the order in which edges are added to the MST.
(c) Run Dijkstra’s single-source shortest paths algorithm on this graph, giving the order in which edges are added.

4. Given the following sets in the disjoint set data structure, show the data sets after a call to union(1,2). Assume that we use union-by-size and path-compression. SETS WOULD GO HERE

5. Consider the following graph $G$ given in outgraph notation:

$$G = (V, E)$$
$$V = \{a, b, c, d, e, f, g, h, i, j\}$$
$$E = \{a : f; \ b : c, h; \ c : h; \ d : h, i; \ e : j; \ f : a; \ g : b; \ h : g; \ i : e; \ j : i; \}$$

(a) Give $G$ in ingraph notation above, to the right of the outgraph notation.
(b) Draw $G$ below, with vertices in the given locations. $d$ represents a discovery time, and $f$ the finish time. (I am giving you the location of each vertex below, in order to make the graphs more readable.)

\[
\begin{align*}
  d[a] &= \quad d[b] &= \quad d[c] &= \quad d[d] &= \quad d[e] = \\
  f[a] &= \quad f[b] &= \quad f[c] &= \quad f[d] &= \quad f[e] = \\
  a &\quad b &\quad c &\quad d &\quad e
\end{align*}
\]

\[
\begin{align*}
  f[j] &= \quad d[f] &= \quad d[g] &= \quad d[h] &= \quad d[i] &= \quad d[j] = \\
  f[f] &= \quad f[g] &= \quad f[h] &= \quad f[i] &= \quad f[j] =
\end{align*}
\]

(c) Run DFS on $G$. Assume that the main loop of DFS goes through vertices in the given order (which is alphabetical), and that within each node, neighbors are visited in the given order (which is alphabetical) for each node.

i. Give discovery and finish times for each node in the locations given above.

ii. Label each edge in $G$ by “T” for tree-edge, “B” for back-edge, “F” for forward-edge, or “C” for cross-edge.

(d) Draw $G^T$.

(e) List the strongly connected components in $G$, in the order which our algorithm from class will find them.

(f) Draw the underlying strongly connected component graph for $G$. Label each component by its first alphabetical vertex.

(g) List all topological orderings for your component graph above (as many as you can).

6. Consider that you are running the first Depth First Search of the Strongly Connected Components algorithm we have seen in class. Vertex $u$ has discovery and finish times 1, 10 respectively, and vertex $v$ has discovery and finish times 11, 20 respectively. Knowing nothing more about the graph, when the entire algorithm finishes running, will $u$ and $v$ be in the same Strongly Connected Component, different components, or is there not enough information to know? Explain your answer.

7. Can the following structure occur in calls to the disjoint-set data structure? Assume that union by size and path compression are used. If so, give a sequence of calls to get the structure. If not, explain why not. SOME STRUCTURE GOES HERE
8. Consider the following graph, represented by adjacency matrix.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

(a) To the right of the matrix above, give the recursive parenthesis structure that recursive calls to DFS will make. Assume that where DFS has a choice of vertices to explore, it chooses the lowest numerically named vertex.

(b) Where would DFS topological sort throw an exception (notice an error), what exception should it throw, and why? A brief answer will do.

9. Each of the following show a partial run of a Minimum Spanning Tree algorithm. Solid lines are edges included in the MST, dotted lines are excluded edges, and dashed lines have yet to be determined. For each, tell if the algorithm running is Prim’s, Kruskal’s, either, or neither.

- If it cannot be Prim’s, give a SHORT explanation why not. (One reason will do.)
- If it cannot be Kruskal’s, give a SHORT explanation why not. (One reason will do.)
- If it might be Prim’s mark all possible vertices which might be s.

SOME GRAPHS WITH WEIGHTED EDGES GO HERE. The edges will also be labeled as either being already known to be in the MST, known to be not in the MST, or not yet known either way.

10. For each of the graphs below, run the “best” single source shortest paths algorithm starting with vertex 0. Which algorithm is running, Dijkstra’s, Bellman-Ford, or Topological? Show work, including intermediate distance estimates. Assume that edges are examined in numerical order: for all algorithms, \( i \rightarrow j \) will be considered before \( i \rightarrow k \) if \( j < k \), and for Bellman-Ford, edge \( i \rightarrow j \) will be considered before \( j \rightarrow k \) if \( i < j \). Give your final tree.

SOME GRAPHS WITH WEIGHTED EDGES GOES HERE

11. You are running the Floyd-Warshall all-pairs shortest paths algorithm on a graph with vertices \( V = \{1, 2, 3, 4, 5\} \). You have the following for \( D^0 \):

\[
D^0 = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
0 & -5 & 6 & \infty & \infty \\
2 & 3 & 0 & 2 & \infty \infty \\
3 & \infty & 0 & 5 & \infty \\
4 & \infty & \infty & 0 & 2 \\
5 & \infty & 2 & \infty & 0
\end{bmatrix}
\]

(a) Draw \( G \).

(b) Give \( D^5 \).